Introduction to Arrays

Contents

[I. Array 2](#_Toc199667031)

[1. Introduction 2](#_Toc199667032)

[What does "Linear" mean? 2](#_Toc199667033)

[2. Problems 3](#_Toc199667034)

[Question 1: 3](#_Toc199667035)

# Array

## Introduction

* An array is a **linear data structure** that stores a collection of **elements of the same data type**.
* The elements in an array are **stored contiguously in memory**.
* Key Properties:
  + **Homogeneous**: All elements have the same data type.
  + **Contiguous Storage**: Stored back-to-back in memory.
  + **Indexing**: Elements are accessed using indexes starting from 0.

### What does "Linear" mean?

* **Logically Linear**: Elements follow a sequence — one comes after the other.
* It's **not about memory layout**, but how we access the data.
* Arrays are linear, but trees and graphs are non-linear data structures because elements don't follow a strict sequence.

## Problems

### Question 1:

* Given an array and you want to count the number of elements having at least one element greater than themselves.

Input: [-3, -2, 6, 8, 4, 8, 5]

* + Max element = 8
  + Elements not having any greater than themselves = All elements equal to 8
  + All others have at least one greater value (8)
* Final Answer = Total elements (7) − Count of max elements (2) = 5

#### Step-by-Step Strategy:

* Find the maximum value in the array.
* Count how many times the maximum value occurs.
* Compute answer as: answer = n - count\_of\_max

#### Implementation in Two Loops

1. Step 1: Find the maximum element
2. Step 2: Count frequency of the max element
3. Step 3: Return the answer as ‘n – count’

size\_t Approach1(const vector<int>& vec)

{

int maxElement = INT\_MIN;

for (size\_t i = 0; i < vec.size(); i++) {

if (vec[i] > maxElement)

maxElement = vec[i];

}

size\_t count = 0;

for (size\_t i = 0; i < vec.size(); i++) {

if (vec[i] == maxElement)

count++;

}

return vec.size() - count;

}

#### Single-Pass Implementation

* In a single loop, do both:

1. Track the max element seen so far
2. Maintain its count

size\_t Approach2(const vector<int>& vec) {

size\_t count = 0;

int maxElement = INT\_MIN;

for (size\_t i = 0; i < vec.size(); i++) {

if (maxElement < vec[i]) {

maxElement = vec[i];

count = 1;

}

else if (maxElement == vec[i]) {

count++;

}

}

return vec.size() - count;

}